Project Description

In this project, we are expected to create a kernel module, named ‘reversi.ko,’ that allows the user to play a game of reversi against the CPU. This module is required to be able to track the game board, read and parse user commands, check validity of user commands, and check for game end states. The module must support a specified set of two-digit user commands, respond correctly to those commands, and respond with appropriate errors to others.

Character Device Description

A character device driver is one of two types of device drivers, in Linux, that can be used by the user to access hardware devices. These devices are kernel components that interact with hardware devices, with character devices being used for slower devices that manage smaller amounts of data than those that use block device drivers, the other type of device driver. The user uses these device drivers by utilizing certain system calls, open, read, write, and close being some examples of them. These commands are redirected to the device driver that interacts with hardware devices. Character device drivers use a cdev struct to register itself in the system and file_operations, file, and inode structs to perform operations. The file_operations struct is used to implement file specific system calls to the character driver, with the inode and file structs being used to identify files to the character driver.

Data Storage

I plan on using a 2d character array to implement the game board in this project. The 2d array will consist of 8 arrays of 8 characters in length, each holding a row of the reversi board. This will help ease coding for different algorithms, by allowing me to both print the board row-by-row and use row-column indexes to reference specific cells of the board.

Algorithm Implementation

- Begin a new game — Copy a pre-set ‘initial board’ 2d character array into the game board, then setting the turn as specified.
- Check valid moves — Check validity of each cell, for current player, by placing the user’s piece in each open cell, checking each connected space for a valid piece chain, noting validity of the cell, and removing the user’s piece from that cell. Valid moves will be kept in a separate 2d array, illustrated by copying the user’s piece into that array at the index being noted. This array will be cleared and repopulated after each move.
- Place Piece — Copies user piece into specified index of game board array, checks each direction for valid piece chain, starting from added piece, copies user piece into each valid piece chain, then sets valid move array for next player.
- Return the current state of the game board — Use a loop to print each array in the 2d game board array, followed by a tab, an ’X’ or an ‘O’, and a newline character.
- Specifies a move for the player — Checks if requested move is valid, returning an error, if invalid, and editing the board accordingly, if valid.
- Asks the computer to make a move — Check for valid computer moves, passing the turn if none exist and selecting one if at least one exists, using the same board editing function as above.
- Pass the user’s turn because no valid moves exist — Check number of valid user moves and pass the turn if none exist.

Estimated Code Length

- Begin a new game — 5 lines
- Check valid moves — 35 lines
- Place Piece — 45 lines
- Return the current state of the game board — 5 lines

- Specifies a move for the player — 5 lines
- Asks the computer to make a move — 5 lines
- Pass the user’s turn because no valid moves exist — 5 lines

Citations

Băluță, Daniel, et al. “Character Device Drivers¶.” *Character Device Drivers - The Linux Kernel Documentation*, Computer Science and Engineering Department, the Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest., linux-kernel-labs.github.io/refs/heads/master/labs/device_drivers.html#data-structures-for-a-character-device.

Sebald, Lawrence. “Principles of Operating Systems.” *CMSC 421 - Project 3,* Lawrence Sebald, 29 Mar. 2021, www.csee.umbc.edu/courses/undergraduate/421/spring21/project3.html.